(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 287 301 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 88303218.7

(51) Int. Cl.⁴: G06F 13/40

(22) Date of filing: 11.04.88

(30) Priority: 17.04.87 US 40513

(43) Date of publication of application:
19.10.88 Bulletin 88/42

(84) Designated Contracting States:
DE FR GB IT SE

(71) Applicant: TANDEM COMPUTERS INCORPORATED
19333 Vallco Parkway
Cupertino California 95014(US)

(72) Inventor: Burnick, David Lee
6249 Gushee Street
Felton California 95018(US)
Inventor: Chan, Kenneth Kin
828 Foxfidge Way
San Jose California 95133(US)
Inventor: Chan, Wing Ming
4080 Merganser Drive
Fremont California 94536(US)
Inventor: Dan, Yie-Fong
3368 Americus Drive
San Jose California 95148(US)
Inventor: Hoang, Duc Manh
3055 Silver Estates
San Jose California 95135(US)
Inventor: Hussain, Zubair
3533 Rollingside Drive
San Jose California 95148(US)
Inventor: Iswandhi, Geoffrey Igantius
1082 Susan Way
Sunnyvale California 94087(US)
Inventor: Korpi, James Edward
3725 Prunteridge Avenue
Santa Clara California 95051(US)
Inventor: Sanner, Martin William
107 Sharon Court
Los Gatos California 95030(US)
Inventor: Zwagerman, Jay Allyn
13654 Lexington Court
Saratoga§California 95070(US)
Inventor: Silverman, Steven Gary
3131 Homestead Road
19E Santa Clara California 95051(US)
Inventor: Smith, James Emerson
387 Vasquez Avenue
Sunnyvale California 94086(US)

EP 0 287 301 A2

(74) Representative: Ayers, Martyn Lewis Stanley et al
J.A. KEMP & CO. 14 South Square Gray's Inn London, WC1R 5EU(GB)

(54) **Input/output system for multiprocessors.**

(57) In a digital computer system which employs a plurality of host processors, at least two system buses and a plurality of peripheral input/output ports, an input/output system is provided whereby ownership of the input/output channels is shared. The device controller employs a first port controller having a first ownership latch, a second port controller having a second ownership latch, a first bus, a dedicated microprocessor having control over the first bus (the MPU bus), a second, higher-speed bus, a multiple-channel direct memory access (DMA) controller which is a state machine which controls the second bus (the data buffer bus), a bus switch for exchanging data between buses, a multiple device peripheral device interface, namely a Small Computer System Interface (SCSI), and at least provision for interface with data communication equipment (DCEs) or data terminal equipment (DTEs). The DMA controller arbitrates data bus usage and can allocate alternate bus clock cycles in response to requests to exchange data and is capable of supporting overlapping transfers. The microprocessor is allowed access to the data buffer bus only if the data buffer bus is not in use for data transfer. The latches associated with each port grant ownership to either port or both ports allowing data exchange between addressed peripheral devices and requesting ports.

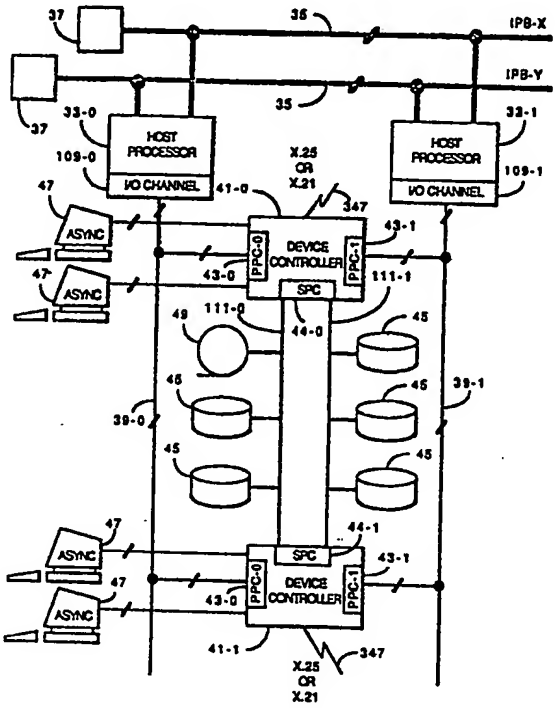

FIG. 1

INPUT/OUTPUT SYSTEM FOR MULTIPROCESSORS

BACKGROUND OF THE INVENTION

The present invention relates generally to communication between a plurality of digital processors in a multiprocessor-based computer system and peripheral devices, such as disk drives and tape drives, which operate at disparate speeds. More particularly the invention relates to an architecture for communicating between two independently-operating ports and peripheral devices coupled to a version of the Small Computer System Interface (SCSI).

Several classes of digital computer systems use a plurality of independent processors to perform computer operations. Examples include fault-tolerant, modular and parallel processing systems. These processors, herein referred to as host processors, require digital data exchange (input and output) with other devices, known as peripherals.

Typically data is exchanged via a system bus under the supervision of a bus controller. Information exchanged with the peripheral devices is typically under the supervision of an input/output subsystem called a device controller.

In one example of a multiprocessor-based fault-tolerant computing system disclosed in U.S. Patent No. 4,228,496 to Katzman et al., a device controller is provided which facilitates exchange of high speed, essentially synchronously-communicated data with relative low-speed, essentially asynchronous peripheral devices. The device controller is constructed to insure that no single host processor failure can impair system operation. The Katzman et al. patent is incorporated herein by reference and made a part hereof, since a specific embodiment of the present invention is intended to improve upon the device controller disclosed in the Katzman et al. patent.

The device controller in the Katzman et al. patent is constructed with the intention of handling exchange of data at relatively high data rates as far as the host processor is concerned in order to minimize interference with programs running in the host processors. The device controller is an interrupt-driven system and provides an input/output program interrupt (IO interrupt) only upon completion of a data transfer in order to relieve the host processor of the burden of being dedicated to the peripheral device during data transfer. The device controller has multiple ports, each of which is logically and physically independent of all other ports so that each device controller can be connected for access to at least two different host processors.

To accomplish its tasks, the device controller of the computing system disclosed in the Katzman et al. patent employs a microprocessor which is dedicated to input/output operations (I/O MPU). The I/O MPU controls an input/output bus structure internal to the device controller (device controller bus structure) over which is carried all data from any requesting host processor and any responding peripheral device as well as data intended solely for its own operation and control of commands. It is through this internal bus structure that the host processor informs the I/O MPU of data transfer type, requested peripheral device, data block size, priority and the like. It is also through the I/O MPU and the device controller bus structure that "ownership" of the operation of the device controller is effected.

Experience with the device controller in accordance with the patent has uncovered a number of shortcomings. For example one shortcoming is within the device controller itself. It has been discovered that, despite the advantage of independent control of the input/output functions by a dedicated microprocessor, such an arrangement does not make the most efficient usage of the valuable time of the host processor. It has been found that even the fastest microprocessors still require a finite time to interpret commands (e.g., input/output requests), which does not make efficient use of time on both the device controller bus structure and on the system bus in the relatively complex environment of a multiprocessor system.

Moreover, in the device controller of the Katzman et al. patent, the device controller is constructed to grant "sole ownership" to only one host processor at a time through a switching scheme which selects among ports of the device controller. Conventionally, there are only two ports and only one ownership latch which grants ownership to either a first port or a second port and hence only one channel I/O bus. Hence, even though there are in theory multiple paths to a peripheral device through a device controller, only one peripheral can be used by a host processor, and only one host processor can have access to any peripheral through the device controller when but a single port is in use and in the process of exchanging data. Access to the device controller from other host processors must thereafter be made through the host processor having ownership of the device controller. This represents a serious shortcoming when several peripheral devices are coupled through a single device controller and several host processors desire access to those peripheral devices.

Another shortcoming of the device controller in accordance with the invention disclosed in the Katzman et al. patent is the limited ability to connect the device controller with large numbers of peripheral devices or peripheral devices having an interconnection structure of selected types which have been widely accepted as standards. With the proliferation of peripheral device types, it is no longer feasible to construct a dedicated interface for each type or brand of peripheral device, each with its unique requirements and characteristics. This is particularly true at the growing "low end" of the computer market, where systems once costing hundreds of thousands of dollars are being superseded by system costing only a fraction of its predecessor. One reason has been standardization, which creates economies of scale.

However, there has been little done to merge fault-tolerant computing systems with their advantages with the various standard input/output interfaces or buses. One emerging peripheral device input/output interface is the SCSI bus, which is now widely accepted for many personal computer applications. As a consequence, many low-cost yet high-quality peripheral devices are now available.

What is needed is a device controller which is not subject to these shortcomings.

SUMMARY OF THE INVENTION

According to the invention, in a digital computer system which employs a plurality of host processors, at least two system buses and a plurality of peripheral input/output ports an input/output system is provided whereby ownership of the input/output channels is shared. The device controller includes a first port controller having a first ownership latch, a second port controller having a second ownership latch, a first bus, a dedicated microprocessor having control over the first bus (the MPU bus), a second, higher-speed bus, a multiple-channel direct memory access (DMA) controller which is an extremely fast state machine, which controls the second bus (the data buffer bus) and access to the input-output ports, a bus switch whereby data communicated between devices on the MPU bus and devices on the data buffer bus are exchanged efficiently, a multiple port peripheral device interface, such as a Small Computer System Interface (SCSI), and at least provision for interface with data communication equipment (DCEs) or data terminal equipment (DTEs). The data buffer bus operates in a synchronous manner. The DMA provides dynamic arbitration of the use of channels of the DMA. The DMA allocates alternate bus clock cycles in response to requests to exchange data. The microprocessor is allowed access to the data buffer bus only if the data buffer bus is not in use for data transfer.

In a specific embodiment, the data buffer bus has coupled to it all input/output ports to the system bus, the DMA controller, a high-speed buffer memory, the bus switch, and a multiplexed SCSI controller, and the MPU bus has coupled to it a microprocessor, program memory, data memory, command registers, communications devices (such as interfaces for data terminal equipment and data communication equipment), and the bus switch. All data is routed through the data buffer bus. The microprocessor configures the SCSI controller and the DMA controller for the type of data exchange. The latches associated with each port grant ownership to either port or all ports allowing data exchange between addressed peripheral devices and requesting ports. Ownership is not surrendered so long as a host processor has need of a device controller.

The invention will be better understood by reference to the following detailed descriptions taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram view of a multiprocessor system constructed in accordance with one embodiment of the present invention, including a plurality of dual-port device controllers sharing peripheral devices coupled to Small Computer System Interface buses.

Figure 2 is a simplified block diagram of one embodiment of a device controller according to the invention illustrating a direct memory access controller, a data buffer bus, a microprocessor bus, a bus switch and two SCSI buses.

Figure 3 is a simplified block diagram one embodiment of a dual-channel DMA controller.

Figure 4 is simplified state diagram of operation of one embodiment of a dual-channel DMA controller.

Figure 5 is block diagram of the architecture of a specific embodiment of a dual-channel DMA controller.

Figure 6 is a timing diagram illustrating one type of operation of a specific embodiment of a dual-channel DMA controller.

Figure 7 is a block diagram of a specific embodiment of two peripheral port controllers according to a specific embodiment of the invention.

Figure 8 is a block diagram of a specific embodiment of a Small Computer System Interface controller according to the invention.

Figure 9 is a block diagram illustrating a specif-ic embodiment of the relationship between software modules according to the invention.

Figure 10 is a block diagram illustrating a specific embodiment of calling protocols between software modules according to the invention.

Figure 11 is a block diagram of a bus switch in accordance with a specific embodiment of the invention.

DESCRIPTION OF SPECIFIC EMBODIMENTS

Reference is made to Figure 1 for an overview of a specific embodiment of the invention. In Figure 1, several processor modules, hereinafter host processors 33 (also designated 33-0 and 33-1), are connected by dual interprocessor buses (IPBs) 35, an X bus (IPB-X) and a Y bus (IPB-Y). Each bus 35 is controlled by a bus controller 37. Each host processor 33 is provided with an input/output channel (I/O channel) 109 (designated 109-0 and 109-1) coupled to a channel I/O bus (CIO bus) 39 (designated 39-0 and 39-1).

Figure 1 also shows a plurality of device controllers 41 (designated 41-0 and 41-1) connected in accordance with the invention. The CIO bus 39 of each host processor 33 is coupled to preferably at least one port of each device controller 41, and more specifically, first CIO bus 39-0 is connected to a first peripheral port controller (PPC-0) 43-0 of each device controller 41-0 and 41-1 and second CIO bus 39-1 is coupled to a second peripheral port controller (PPC-1) 43-1 of a each device controller 41-0 and 41-1.

The device controller 41 is provided with a protocol controller (SPC) 44 for two peripheral device buses 111 of a standardized configuration known as the Small Computer System Interface (SCSI), designated herein SCSI-0 111-0 and SCSI-1 111-1. The SCSI standard has been described in publications by the American National Standards Institute as ANSI Standard X3.131-1986, and commercial component parts are available for implementing functions of the SCSI bus. For example, in a specific embodiment of the invention, the SCSI interface is implemented by a type MB87030 chip made by Fujitsu and available from Fujitsu America of Santa Clara, California.

The primary function of the SCSI buses 111 is to provide port connections for standard SCSI peripherals, such as a tape drive 49 (with internal SCSI controllers) and disk drives 45 (with internal SCSI controllers). The SCSI bus is of a type which allows more than one device controller to be connected thereto. (For example, up to eight units or devices, including device controllers, can be connected to a single SCSI bus.) Hence, both the first device controller 41-0 and the second device controller 41-1 are connected to the first SCSI bus 111-0, the first SCSI bus 111-0 having connected thereto a first set of SCSI peripheral devices 45 and 49. In addition, the first device controller 41-0 and the second device controller 41-1 are also connected to the second SCSI bus 111-1, the second SCSI bus 111-1 having connected thereto a second set of SCSI peripheral devices 45. Thus each CIO bus 39 has at least two paths to each SCSI device 45 or 49, and each host processor has at least four paths (two through another host processor via an IPB 35) to each SCSI device 45 or 49. This configuration substantially enhances the reliability of the overall computer system and permits the temporary removal for example of one device controller without compromising access to any other of the SCSI devices.

As a further feature, the device controllers 41 may also provide for access to or from the host processors 33 for asynchronous terminals 47 and packetized communication links 347 including links conforming to the ANSI protocol standards X.25 or X.21.

The configuration and computer architecture represented by Figure 1 is a new type of system representing a departure and enhancement over the structure disclosed in the Katzman et al. patent, incorporated herein by reference.

Referring now to Figure 2, there is shown a block diagram of the device controller 41 in accordance with a specific embodiment of the invention. According to the invention, there is provided in combination a data buffer bus (DB bus) 100 and a microprocessor bus (MPU bus) 102 coupled to one another by means of a first bus switch (DBBS) 104. In accordance with the invention, the DB bus 100 is a synchronous bus whose operational clock rate is a substantially different rate than he MPU bus 102, and the DBBS 104 is a bus switch which buffers transfers of data and commands between the DB bus 100 and the MPU bus 102 to assure timely transfer of data and commands between the buses operating at disparate clock rates.

Coupled to the DB bus 100 are a data buffer direct memory access controller (DB DMA) 106, the first PPC 43-0, the second PPC 43-1, the SPC 44 and a high speed read/write random access memory (DB MEM) 108 for temporary storage of all data-type information transferred over the DB bus 100.

Coupled to the MPU bus 102 is a microprocessor (MPU) 119 (such as a type 68010 microprocessor made by or under license to Motorola), instruction memory such as Electronically Erasable Read Only Memory (EEROM) 110 in the amount of 16 KB, random access memory (MEM) 112 in the amount of 512 KB, a conventional direct memory access controller (DMA) 114 such as a type 68450 DMA device made by, e.g., Hitachi (part HD68450), and a standard serial communication chip device

(SCC) 116 (such as a Zilog Z8030 serial communication chip) supporting both X.25 bit-oriented packet protocol and X.21 byte-oriented packet protocol, at least one standard dual universal asynchronous receiver-transmitter (DUART) 118-1 and 118-2 such as a Signetics type 2861 chip made by Signetics. The external configuration of the SCC 116 and the DUARTs 118 are device dependent and so require no further explanation.

The DMA 114 is used to transfer data to and from the SCC 116 or the DUARTs 118 and the MEM 112 on the MPU bus. It cooperates closely with the MPU 119. DMA 114 channels can be programmed to move large data blocks and generate an interrupt signal to the MPU 119 at the end of a transmission.

The SPC 44 has two ports 144-0 and 44-1 for SCSI buses 111-0 and 111-1. The SCSI buses are multiplexed. In addition, as explained hereinafter, an SPC bus 120 is provided as part of the SPC configuration together with SPC bus switch 122, which provides bi-directional command and control coupling between the MPU bus 102 and the SPC bus 120.

The device controller 41 generates or responds to a plurality of interrupts on interrupt lines (not shown for clarity). The interrupts are generated by I/O channels through the PPCs 43, by I/O devices through the SPC 44, by DMA channels from the DB DMA I08 or DMA 114, by communication interfaces such as the SCC 116 and the DUARTs 118, by timeouts and by hardware errors. Of primary interest is the interrupt issued by the PPCs 43 in response to an EIO from a host processor.

With reference to Figure 3, the DB DMA 106 is a very fast state machine whose function is to control access to the DB bus 100. The DB DMA 106 employs a technique known as dynamic arbitration to allocate time for data transfer on the DB bus 100 between the PPCs 43 and the peripheral devices through the SPC 44. To this end, the DB DMA 106 includes arbitration logic 200 for allocating time on the DB bus 100 (Figure 2) between two channels of data transfer functions as well as device controller management functions carried out under control of the MPU 119, all of which require usage of the DB bus 100. The DB DMA 106 is a dual channel device which is capable of concurrent support for two channels of data, of being reprogrammed to support chaining of consecutive frames of data without having to repeat an interrupt cycle for each frame of data, and of being reprogrammed to support data transfer at twice the standard rate by use of both channels for a single task.

Referring to Figure 3, Figure 4 and Figure 6 together, the arbitration logic is responsive to the following requests: Request for Channel 0 (REQ0*) (the * is used to denote logic LOW is TRUE), Request for Channel 1 (REQ1*), and bus request for the MPU 119 (BREQ*). The DB DMA issues at least the following interrupts: Acknowledge availability of Channel 0 (ACK0*), Acknowledge availability of Channel 1 (ACK1*), Bus usage granted in response to BREQ* (BGNT*). A sample of the relative timing of each of these interrupts is shown in Figure 6 in relation to a system clock (SYSCLK) and two-channel data transfer (XFR0 and XFR1). Figure 6 illustrate operation of the DB DMA 106 if all request signals are asserted simultaneously. First a three-way conflict is illustrated and then a two-way conflict between channels is illustrated. Data transfer via a single channel is permitted to occur no more often than one-half of the available time. However, the DB DMA 106 via the arbitration logic 200 exercises such control over the DB bus 100 that virtually no clock cycle is wasted if there is data to be transferred or a command to be processed. The maximum latency (210 Figure 6) from DMA Request to DMA Acknowledge is two clock cycles for either channel of the DB bus 100. (The latency time for a bus request depends on the load on the channels. There may be a timeout provided for example. In the extreme, the latency time can be no longer than the time required to exhaust the queue of outstanding channel requests, since all channel requests are generated by the MPU 119.)

Referring again to Figure 3 and Figure 4, the arbitration logic is in communication with three substantially identical state machines, a channel 0 state machine (CH0 SM) 202, a channel 1 state machine (CH1 SM) 204 and a bus request/bus grant state machine (BREQ/BGNT SM) 206. Each of the state machines 202, 204, 206 is substantially identical in that each has four states S0, S1, S2, S3 as represented in Figure 4. State S0 is the idle state; state S1 is the acknowledge state following any request made (REQ0, REQ1, BREQ) during the idle state; state 2 is a data transfer state; and state 3 is the error state. When either or both data transfer channels are inactive (CH0 SM 202 and/or CH1 SM 204 are in the idle state S0) and the BREQ* signal is asserted, then the DB bus 100 is relinquished to the MPU 119 for one transfer period (e.g, BXFR 212 Figure 6). When any of the state machines 202, 204, 206 is in the idle state S0, assertion of a Request signal (REQ*) causes transition to state S1. When the state machine is in state S1, the DB DMA 106 is computing the address to be used by the DB MEM 108 as well as address parity for the next address. State S1 is the Acknowledge state, meaning that the state machine expects a certain sequence of signals following issuance of an acknowledgement to the requesting element. Specifically, continued assertion of a Request signal (REQ*) and transfer of data is ex-

pected following state S1; otherwise an error condition is indicated (state S3). The state machine transitions between state S1 and state S2 until all data transfer is complete and the Request signal is withdrawn (REQ* disasserted). When in state S1 REQ* is disasserted, the state machine transitions to the Error state S3. It remains in the "Error" state until assertion of the Reset signal. When the Reset* signal is asserted in the Error state, the state machine is restored to the Idle state S0. When the state S2 is invoked, data transfer takes place; hence its designation as the Data transfer state (DXFR). If the request signal (REQ*) remains disasserted, the state machine reverts to state S1. If no new Request signal is asserted before a Reset signal (RESET*) is asserted, the state machine returns to the Idle state S0. If however in state S2 the request signal REQ* is disasserted, the state machine also returns to the Idle state S0.

Each of the channels for data transfer (XFR0 and XFR1 of Figure 6 referring to CH0 and CH1, respectively) operate independently of one another but of course cannot, during the same bus cycle, transfer data between the same peripheral device through the SPC 44 and the same requesting port PPC-0 43-0 or PPC-1 43-1. Every normal data transfer is written to or read out of a memory location in the DB MEM 108 which has been designated by the memory address output (MADRS) 224 of the DB DMA (Figure 5).

Referring to Figure 5, there is shown a diagram of the architecture of the DB DMA 106. A key feature of the architecture is a fast adder or more specifically a two level Carry Look Ahead (CLA) adder which is capable of doing add and subtract operations for the channel address generation in less than one clock period. The contents of channel 0 and channel 1 address registers 216 and 218 are provided as one input to the CLA adder 214 via an address multiplexer ADRS MUX 220. The augend address (ADRS AUGEND) 222 is provided as the other input to CLA adder 214, the output of which is the memory address (MADRS) in the DB MEM 108 to or from which the data on the DB bus 100 is written or read.

The DB DMA 108 is capable of chaining related data in a block for continuous transfer of priority data at the designated standard data transfer rate. This is a function which would normally be carried out by the MPU 119 (Figure 2) and would require that an interrupt cycle be repeated for each frame of data. However, the DB DMA 106 according to the invention is instead employed for this purpose. The result is a substantial increase in speed of selected data transfer operations of the device controller 41.

Referring to Figure 5, a Master Control register 260, a Channel 0 control register 262, and a Channel 1 control register 264 are coupled to receive input data via a register bus 266 from a data input/output port 268 and further to provide output to a the register bus 266 and the data input/output port 268 via a register multiplexer 270. Further as shown in Figure 5, and now referring to the mode of operation as well as the architecture, a Channel 0 chain pointer register CH0 CHAIN PTR REG 226 is coupled to Channel 0 address multiplexer 228 to the Channel 0 address register CH0 ADR REG 216, and Channel 1 chain pointer register CH1 CHAIN PTR REG 230 is coupled to Channel 1 address multiplexer 232 to the Channel 1 address register CH1 ADR REG 218. The chaining function is enabled by placement of a suitable command word in the channel control register 262. Once chaining has been enabled, the respective channel address registers 216 and 218 of the enabled channel are loaded from the corresponding channel chain pointer register 226 or 228, so that chained sequencing is started. Chained sequencing continues until chain parameters are fetched from memory; thereafter regular channel operation resumes. The ability to reprogram a channel of DB DMA 106 for chaining operations while supporting normal functions on other channels represents a departure from known technology in this field.

In order to check for completion of each transfer, a comparator 250 is provided for comparing the content of the active channel address register 216 or 218 via the address MUX 220 with the content of the corresponding end pointer register 254 or 256 via an end address multiplexer (END-ADRS-MUX) 258. The output of the comparator 250 is either an Equal or NOT Equal value, which is used to control data transfer.

Another feature of the invention is the use of address parity prediction to assure that the system is operating properly. More specifically, address parity of the next scheduled address is predicted and compared with actual parity to assure that data is actually being progressively transferred to and from the proper memory locations. To this end a conventional parity generator 270 is provided having as its input the output of the CLA adder 214, a parity predictor 272 having as its input the output of the channel address register 216 or 218 via the output of the address multiplexer 220 and as its other input selected control bits of the corresponding channel control register 262 and 264 via the multiplexer 270 through register bus 266. The outputs of the parity generator 270 and the parity predictor 272 are coupled to a comparator 274. The channel control register 262, 264 contains information regarding the type of transfer and the address increment for each channel. The channel address register 216, 218 contains the current address of each channel. The parity predictor 272

comprises means for combining the address increment and the current address to obtain a predicted parity. The output of the comparator 274 is thus an indication of whether the current parity and the parity predicted from the previous address. Inequality is an error condition.

Figure 7 is a block diagram of a specific embodiment of two peripheral port controllers (PPCs) 43-0 and 43-1 according to a specific embodiment of the invention. According to the invention there is associated with or incorporated into each PPC 43 an independently operated PPC latch 300 (PPC-0 has PPC latch 300-0 and PPC-1 has PPC latch 300-1). Each PPC latch 300 operates independently so that two I/O channels can have ownership set simultaneously. In a specific embodiment as shown in Figure 7, each PPC 43-0, 43-1 includes a port chip 302-0, 302-1 coupled to a data buffer 303-0, 303-1 whereby EIOs issued on the CIOs 39-0, 30-1 cause interrupts and set the PPC latches 300-0, 300-1. The port chips 302-0, 302-1 also provide limited data buffering between the CIOs 39-0, 39-1 and the DB bus 100. Each PPC 43 has the following registers for receiving EIOs and data buffering: Buffer_A: a data buffer
Buffer_B: a data buffer
Buffer_C: a data buffer
RAC: register for storage of the Read Address and Command word of a reconnect sequence
RIC: register for storage of the Read Interrupt Cause word of an interrupt sequence
RIST: register for storage of the Read Interrupt STatus word of an interrupt sequence
LAC: register for storage of the Load Address and Command word of an EIO sequence
LPRM: register for storage of the Load PaRameter word of an EIO sequence
Status: three registers providing access by the MPU 119 to status signals within the PPC
Flag Control: register whereby the MPU 119 can set and clear various flags in the PPC
Burst Length: register controlling the maximum number of word in a single reconnect burst (maximum 128).

The DB DMA 106 has direct access to the three Buffers A, B and C for both read access and write access. These buffers are word-wide registers configured in a bidirectional queue. These registers are clocked by the system clock (SYS_CLK) (associated with the host processors) during transfers in from the DB bus 100 to the CIO 39. During out transfers they are clocked by the CIO 39 channel handshake signal SVO (see the Katzman et al. patent for further explanation of the CIO 39). Two three-bit circular shift registers are used as pointers. The first pointer is clocked by SYS_CLK, and the shift is enabled by the DMA_ACK signal from DB DMA 106 (Figure 6). The second pointer is clocked by SVO, and shift is enabled by an IN or OUT TBUS command through the CIO 39. The two pointers keep track of the next register (Buffer A, B, or C) to be loaded with data or unloaded. A two-bit synchronous counter keeps track of the number of words in the buffers A, B, and C. The synchronous count is used to determine when the DMA_REQ" signal must be asserted, and it is available to the MPU 119 via the status registers.

In the idle state the PPC latches 300 do not assert a "Port Own" signal (PO_OWN" by PPC latch 300-0 and P1_OWN" by PPC latch 300-1). However, whenever the appropriate EIO is received by the PPC chip 302, a "take ownership" interrupt signal is asserted (Takeown") by the PPC chip 302 to the PPC latch 300 via signal line 304. Each PPC port 41 is properly identified via comparators 306 and appropriate switches 308. The addressed PPC latch 300 asserts port ownership (PO_OWN" and P1_OWN") as interrupts to the MPU 119. It is thereafter the task of the MPU 119 to interpret the EIO, to designate which of the ports is to have which data channel of the DB bus 100 and to configure the DB DMA 108 to so allocate the two data channels CH0 and CH1 of the DB bus 100 to one or both of the designated ports. Because of dual ownership through the PPC latches 300 and dual channel operation of the DB DMA 106, the device controller can handle all of the protocols and data transfers to and from peripheral devices within a common time frame and as far as the CIO buses 39 are concerned, simultaneously. Each PPC 43 is capable of essentially simultaneous operation. In simultaneous operation, a first PPC 43-0 can actively transfer data and commands while the second PPC 43-1 can also actively transfer information; however such transfer is limited to commands.

All of this is done without direct data transfer via the MPU 119, which represents a substantial departure from other device controllers known in this field, i.e., device controllers which can be reconfigured in accordance with the type of data transfer desired.

In the process of receiving an EIO, the DB DMA 106 must grant bus access to the MPU 119. To this end, the MPU 119 responds to the appropriate interrupt issued by the PPC 43 by issuing a BRQT signal to the DB DMA 106 via the bus switch 104.

Referring to Figure 11, there is shown one embodiment of the bus switch 104. The bus switch 104 comprises a three-state buffer 400, a delay means 402 and a latch 404. The output enable terminal of the three state buffer 400 is controlled from the DB DMA 106 (a clock line synchronized with the channel in use) through the delay means

402 whose function is to enable the three-state buffer 400 after sufficient time has elapsed following a prior transfer of data to minimize interference between consecutive data blocks from the MPU bus 102 to be transferred to the higher speed DB DMA bus 100. The delay means 402 typically provides a 20 ns delay and therefore a 20 ns window between data transfers between data transfers operating on a 200 ns clock. The bus switch 104 further comprises a latch 404 which is coupled to receive data from the higher speed DB bus 100. It operates to latch the received data for use by the slower MPU bus 102. The latch 404 is also a three state device wherein the output enable terminal is under control of the MPU 119, since the MPU 119 decides when to read the contents of the latch 404. Significantly, operational control of the bus switch 104 resides in the DB DMA 106, and more particularly is part of the arbitration logic 200 and Bus Request/Bus Grant protocol of the BREQ/BGNT state machine 206 for communication with MPU 119 and any other element coupled to the MPU bus 102. (Most data transfer for example is via the memory 112 under control of the 68450 DMA 114.) Operation of the bus switch 104 in conjunction with the MPU 119 is illustrated by the following example.

The transfer of the EIO to the MPU 119 is from the PPC 43 through the bus switch 104 to the MPU 119. The PPC 43 issues an interrupt to the MPU 119 which in turn initiates a Bus Request to the DB DMA 106. The DB DMA 106 then issues a Bus Grant to the MPU 119. The MPU 119 then captures the EIO from the PPC 43 via the DB bus 100 and the bus switch 104. The MPU 119 then interprets the command structure while the DB DMA 106 controls other data transfers on the DB bus 100. Once the MPU 119 has completed interpretation of the EIO it again requests a bus cycle of the DB bus to access the DB DMA 106 to instruct it regarding the port and form of data transfer. Once a Bus Grant is given the MPU 119 transfers instructions to the registers of the DB DMA 106 via the bus switch 104.

One of the significant advantages of the multiple-bus architecture of the invention wherein the synchronous buses are coupled by bus switches which comprises buffers and latches is that different speed devices can be interfaced very easily and even changed if necessary. For example, if it is desired to substitute a faster microprocessor chip in the position of MPU 119, it becomes the relatively trivial task of exchanging components and boosting MPU bus clock speed. The speed and operation of the DB bus 100 is not affected.

The MPU 119 accesses the SPC 44 via the SPC bus switch 122 and SPC bus 120 in order to configure the SPC 44 for the data transfer designated by the EIO and related vectors.

Referring to Figure 8, there is shown the SPC 44 in accordance with the invention. The SPC 44 is built around a SCSI Protocol Controller Chip (SPC Chip) 402 which in a specific embodiment may be a type MB87030 supplied by Fujitsu. The SPC Chip 44 is coupled indirectly to the DB bus 100 and directly to the SPC bus 120. Coupled between the DB bus 100 and the SPC chip 402 is a byte assembly register 404 on the DB bus 100, which assembles the data bytes for the SPC Chip 402 to be passed via lines 403, and a parity register 406 which creates odd parity for data in the byte assembly register 404 to provide compatibility with the parity expected by the SPC Chip 402.

The SPC bus switch 122 comprises a latch 410 and a buffer 412. It is coupled between the MPU bus 102 and the SPC bus 120 to provide bi-directional command and control coupling between the MPU bus 102 and the SPC bus 120. The SPC bus switch 122 is enabled by a signal on interrupt line 408 from the MPU 119 to latch 410 and buffer 412.

Since the SPC Chip 402 has only one SCSI port, provision must be made according to the invention to connect two SCSI buses 111-0 and 111-1. Therefore SPC Chip 472 provides a set of driver lines 414 and receiver lines 416 to a multiplexer set 418. In the illustrated embodiment, the multiplexer set 418 for the SPC Chip 402 itself comprises first and second sets of dual input open collector output NAND gates 420-0 and 420-1 which are functional as enablable drivers for SCSI bus 0 111-0 and SCSI bus 1 111-1, respectively, and corresponding first and second sets of inverters 420-0 and 420-1, which are functional as three-state receivers for SCSI bus 0 111-0 and SCSI bus 1 111-1, respectively. In addition third and fourth inverter sets 424-0 and 424-1 are coupled as receivers from the respective SCSI buses 111-0 and 111-1 and supplied to SCSI interrupt logic means 426, which is coupled to the SPC bus 120 and which has interrupt line 428 which is coupled to the MPU 119. Still further, a bus select register 430 is provided on the SPC bus 120 which has as outputs an Enable Bus 0 line 432 and an Enable Bus 1 line 434. Enable Bus 0 line 432 is coupled to drivers 420-0 and receivers 422-0 associated with SCSI bus 0 111-0, and Enable Bus 1 line 434 is coupled to drivers 420-1 and receivers 422-1 associated with SCSI bus 1 111-1.

The SPC 41 can respond to programming commands from the MPU 119 via the SPC bus 102 directed to the SPC Chip 402, can pass data (read and write) to/from the DB bus 100 from/to either SCSI bus 111-0 or 111-1 as selected by a command from the MPU 119 via the bus select register means 432, or it can interrupt the MPU 119 via

interrupt line 428 from the SCSI interrupt logic means responsive to signals from peripheral devices on either SCSI bus 111-0 or 111-1. Hence, the device controller 41 permits multiple access from multiple channels to multiple devices on selectable SCSI buses within the same operational cycle, during two or more overlapping commands.

To facilitate odd byte transfers on the DB bus 100 to/from the SPC Chip 402, the byte assembly register 404 can be read and written in word mode by the MPU 119 via the DB bus 100. This facilitates data transfer between an eight bit device (the SPC Chip 402) and a high speed sixteen bit bus (DB bus 100). Specifically, during IN mode transfers from the SPC Chip 402, the first byte to the DB bus 100 is stored in the most significant byte side of the byte assembly register 404 and when the second byte is ready, the first byte is passed from the byte assembly register 404 to the most significant byte side of the DB bus 100 while the least significant byte is passed via three state driver 405 to the least significant byte side of the DB bus 100. During OUT mode transfers to the SPC Chip 402, the least significant byte of the DB bus 100 is transferred to the least significant byte side of the byte assembly register 404 as the most significant byte of the DB bus 100 is transferred directly to the the SPC through three state driver 407. The next byte transferred to the SPC Chip 402 is the least significant byte from the byte assembly register 404.

The SCSI interrupt logic means 426 is a logic device which controls three conditions loosely referred to as interrupts from each of the two SCSI buses 111 thereby providing an interrupt on interrupt line 428 to the MPU 119 under six possible conditions. The conditions are: 1) SCSI Bus was Reset; 2) SCSI Bus was Free; 3) Device Controller is being Reselected on the SCSI bus. All six conditions are "OR"ed together so that any one of the conditions will cause the interrupt to issue via interrupt line 428 to the MPU 119.

The six conditions to cause the interrupt can be individually enabled, read and cleared from the MPU 119 by means of six lines to read/write/control port 429 coupled to registers within the SCSI interrupt logic means 426. In addition a master interrupt enable may be provided via a seventh line to read/write/control port 429. These interrupts monitor both of the SCSI buses 111 independently of the SPC Chip 402 and of the multiplexer 418 by means of receivers 424-0 and 424-1. Even if the interrupt line 428 is not enabled, the status of the interrupt registers can be polled by the MPU 119 via the SPC bus 120.

The MPU 119 is used for performing the programming of the SPC 44. More specifically, the MPU 119 sets up queues of instructions for the ports and for individual devices in memory MEM 112 which are communicated to the SPC 44 in order of request. The MPU 119 can be handling several tasks at the same time, such as a read of one device which requires a seek before data can be transferred. The MPU 119 can proceed to execute the instruction queue of another device task for another device while awaiting completion of the seek on the first device. In this manner, the processing power of the MPU 119 is used more efficiently.

Referring to Figure 9, there is shown a block diagram illustrating a specific embodiment of the software structure 900 according to the Invention. The software structure 900 is divided into modules according to function, including supervision, device drivers and input/output tasks. The software structure 900 comprises a kernel 910 or operating system, a peripheral port control driver module (PPC driver) 912, a port task 914 logically coupled between the PPC driver and all other tasks, including but not necessarily limited to a task 916 and closely-related utilities, a tape task 918, packet communication (using X.25 protocol) (X.25) tasks 920, asynchronous communication (async) tasks 922, and remote maintenance interface (RMI) tasks 924.

The disk task 916 and the tape task 918 are logically couple to an SCSI peripheral control (SPC) driver 926. The X.25 tasks 920 are logically coupled to a serial communication control (SCC) driver 928. The async tasks 922 are logically coupled to an async driver 930. The RMI tasks are coupled to a maintenance-diagnostics bus (MDB) driver 932. As it is not necessary to an appreciation of the invention to understand the details of the specific drivers or tasks, only selected drivers or tasks are discussed hereinafter for illustration purposes. It is sufficient to one of ordinary skill in the art to know that remote maintenance is provided via the MDB driver which provides device control to a maintenance-diagnostics bus, that the SCC driver 928 controls a standard SCC chip such as a Zilog Z8030 serial communication chip supporting both X.25 bit-oriented packet protocol and X.21 byte-oriented packet protocol, and that the async driver 930 controls standard dual universal asynchronous receiver-transmitters (DUARTs) such as a Signetics 2861 chip.

The kernel 910 is a standard operating system available as a commercial product for the 68000 family of microprocessors. (The 68000 family is made by or under license from Motorola, Inc.) A specific example of a kernel 910 is the T8120 common kernel software module used with Tandem brand computer systems made by Tandem Computers, Inc., of Cupertino, California.

Referring to Figure 10, and using as an exam-

ple a communications between one host processor (not shown, see Figure 1) via the PPC driver 912 and one associated port (not shown, see Figure 1) and a peripheral device (not shown, see Figure 1) on the SCSI bus (not shown, see Figure 1) accessible via the SPC driver 926 and associated hardware, the function of the drivers is to respond to hardware-generated interrupts (such as an interrupt associated with an Execute Input Output command) (EIO) issued by a host processor (not shown see Figure 1) in order to engage a port task 914 to pass messages (including messages associated with the EIO) to another task such as a disk task 916, instructing the other task 916 to carry out a desired action, such as to call a "read data" or "write data" subroutine, to communicate to the peripheral device (not shown) via a driver, such as the SPC driver 926. In response, when the read or write task is done, in the exemplary embodiment, the peripheral device (not shown) generates another hardware interrupt which is received and interpreted at the SPC driver 926 to generate a signal indicating that the command has been completed. The signal CMD COMPLETE or its equivalent is communicated to the disk task 916, which in turn conveys a message to the port task to issue a reconnect or an interrupt. The port task then makes a subroutine call to the PPC driver 912 to issue the reconnect or the interrupt to the host processor.

Thus, communication between the host processor and the device controller is by means of an EIO addressed to a peripheral device under control of a device controller, and the device controller responds to the host processor by a reconnect burst or an interrupt.

The port task 914 and related software have the following characteristics of interest:

The port task 914 sends the EIO Request message it receives to the device task 916 based on the controller address. If the host processor 33 issues multiple EIOs to the same controller address, the port task 914 will send multiple messages to the same device task.

The port task 914 does not associate EIOs with reconnects or interrupts. All decisions to reconnect and interrupt are made by the device tasks.

The port task 914 does not have the concept of ownership. If an ownership concept is necessary, it is provided by the host processor. As far as the device controller itself, the device controller can be owned by both channels simultaneously. Thus all EIOs having the same controller address from either channel are directed to the same device task.

The port task 914 receives the address of each data block from the device tasks and then breaks the data block into bursts to be sent or receive. If the address of the data block is in the memory MEM 112, it will move the data between MEM 112 and DB MEM 108 by using one of two reserved 256 data buffers in the DB MEM 108, moving data into one buffer while reconnecting to the host processor via the PPC on the other buffer.

The port task 914 times the reconnect and interrupt requests. If the PPC 43 does not complete in time, the port task 914 resets the PPC 43 and returns to the the device task with an error status.

Events such as Takeown EIO, Kill EIO and Channel Reset are reported as status to Reconnect, Interrupt and Read_Status and are not sent as messages or broadcast to the device tasks. Channel Reset causes the device controller 41 to reset to its idle state. Takeown EIO requires no action but indicates that a transfer has been aborted.

The port usage operates within the following constraints:

EIOs will be accepted from either port at all times.

A host processor interrupt will be issued only when a port is not in the reconnect condition.

Data for the main memory of the host processor is sent or received subject to availability of memory space in the DMA data buffer (DMA DB MEM 108).

An exemplary listing of device tasks and drivers used in accordance with the invention is provided in Appendix A accompanying this application. However, the operation of the invention would be clear to one of ordinary skill in this art from the foregoing description.

The invention has been explained with reference to specific embodiments. Other embodiments will be apparent to those of ordinary skill in this art in light of this disclosure. It is therefore not intended that this invention be limited except as indicated by the appended claims.

Claims

1. For use in a digital computer system, a device controller comprising:

a first port-input/output controller means coupled to a first input/output channel bus and a second port-input/output controller coupled to a second input/output channel bus;

said port-input/output controller means having:

a first ownership latch means for granting ownership of said device controller to a first host processor, and

a second ownership latch means operative independently of said first ownership latch means for granting ownership of said device controller to a second host processor independently of said first port input/output controller, thereby to provide mul-

tiple data paths between one of said first or second host processors and any peripheral devices coupled to said device controller.

2. The device controller according to claim 1 further comprising:

a peripheral device-input/output controller means; and

multiplexer means coupled to said peripheral device-input/output controller means on one end and coupled to a first SCSI bus and to a second SCSI bus on the other end thereby to provide for connection to either said first SCSI bus or to said second SCSI bus through said device controller.

3. The device controller according to claim 2 further comprising:

a first bus operative at a first clock rate;

a dedicated microprocessor means having control over said first bus (the MPU bus);

a second synchronous bus operative at a second clock rate, different from said first clock rate;

a multiple-channel direct memory access (DMA) controller means comprising a state machine, said DMA controller means having control over said second synchronous bus (the data buffer bus) and control over and access to said first and second input/output channel buses for arbitrating access to said first bus; and

a first bus switch coupled between said first bus and said second synchronous bus for communicating first information on said first synchronous bus at said first clock rate to said second synchronous bus at said second clock rate and second information on said second synchronous bus at said second clock rate to said first bus at said first clock rate under control of said DMA controller and upon request of said MPU.

4. The device controller according to claim 1 further comprising:

a first bus ;

a dedicated microprocessor means having control over said first bus (the MPU bus);

a second bus;

a multiple-channel direct memory access (DMA) controller means comprising a state machine, said DMA controller means having control over said second bus (the data buffer bus) and control over and access to said first and second input/output channel buses for arbitrating access to said first bus; and

a first bus switch coupled between said first bus and said second bus for communicating first information on said first bus to said second bus and second information on said second bus at to said first bus under control of said DMA controller and upon request of said MPU.

5. The device controller according to claim 1 further comprising:

a first bus operative at a first clock rate;

a dedicated microprocessor means having control over said first synchronous bus (the MPU bus);

a second synchronous bus operative at a second clock rate, different from said first clock rate;

a multiple-channel direct memory access (DMA) controller means comprising a state machine said DMA controller means having control over said second synchronous bus (the data buffer bus) and control over and access to said first and second input/output channel buses for arbitrating access to said first bus; and

a first bus switch coupled between said first bus and said second synchronous bus for communicating first information on said first bus at said first clock rate to said second synchronous bus at said second clock rate and second information on said second synchronous bus at said second clock rate to said first bus at said first clock rate under control of said DMA controller and upon request of said MPU.

6. The device controller according to claim 5, wherein said DMA controller comprises a programmable state machine defining by alternating time allocation a first channel and a second channel on said second synchronous bus, said programmable state machine including means for arbitrating time allocation between said first channel and said second channel for data transfer on said second synchronous bus, and for allocating time on said second synchronous bus to said microprocessor means at a lower priority than for said data transfer.

7. The device controller according to claim 5, wherein said programmable state machine further comprises means imposing a maximum latency between a data request and a data acknowledge issued by, said state machine of no more than two bus cycles on any channel and wherein maximum latency between a bus request issued by said microprocessor and a bus grant issued by said state machine is dependent upon load of said first channel and said second channel.

8. The device controller according to claim 5, wherein said DMA controller further includes means for generating channel address values in less than one clock period for assuring timely generation of channel address values.

9. The device controller according to claim 5, further including register means and program means for programming said DMA controller during operation.

10. The device controller according to claim 9, wherein said DMA controller further includes means for programming chaining of related data blocks for continuous transfer of priority data on one of said channels independent of the other of said channels.

11. The device controller according to claim 10, wherein said DMA controller further includes programmable means providing for direct data transfer without intervention of said microprocessor means.

12. The device controller according to claim 5, wherein said DMA controller further includes means for predicting address parity, means for generating address parity and means for comparing predicted parity with generated parity in order to verify that said DMA controller is generating correct sequential addresses.

13. The device controller according to claim 5 wherein said first bus switch comprises:

latch means for latching data from said second synchronous bus for reading by said microprocessor means coupled to said first bus;

three-state buffer means for buffering data sent from said first bus to said second synchronous bus; and

delay means coupled to an enable input of said three-state buffer means for imposing an interval between successive data transfers via said three-state buffer in synchronism with a selected one of said channels of said second bus means.

14. The device controller according to claim 3 further comprising:

a peripheral device-input/output controller means, said peripheral device-input/output controller means having a controller data bus of a first bit width and wherein said second synchronous bus has a second bit width different from said first bit width;

a byte assembly register coupled between said second synchronous bus and said controller data bus for assembling data in bytes and words for exchange between said controller data bus and said second synchronous bus means.

15. The device controller according to claim 14 further comprising:

a parity converting means coupled between said second synchronous bus and said controller bus for converting parity of data passed between said second synchronous bus and said controller bus.

16. The device controller according to claim 14 further comprising:

multiplexer means coupled to said peripheral device-input/output controller means on one end and coupled to a first SCSI bus and to a second SCSI bus on the other end thereby to provide for connection to either said first SCSI bus or to said second SCSI bus through said device controller;

third bus means coupled to said peripheral device-input/output control means;

programmable interrupt logic means coupled between said third bus means and said first SCSI bus and said second SCSI bus to monitor preselected conditions on each of said first SCSI bus and said second SCSI bus; and

second bus switch means coupled between said first bus means and said third bus means for passing command messages to said multiplexer means to select between said first SCSI bus and said second SCSI bus, to said programmable interrupt logic means for preselecting conditions which will cause an interrupt to said microprocessor means, and to said device-input/output controller means for controlling a selected SCSI bus.

17. The device controller according to claim 5 further comprising means coupled to said first bus means for communicating data between DCEs or DTEs via said second synchronous bus and said first port-input/output controller means or said second port-input/output controller means, including at least a universal receiver-transmitter and a synchronous communication controller.

18. The device controller according to claim 5 further including program means for handling Execute Input/Output Commands issued by each of said host processors, to said microprocessor means, said program means comprising:

a peripheral port control driver means and port task means for controlling said first port-input/output controller means or said second port-input/output controller means;

device task means in communication with said port task means; and

device driver means for controlling peripheral devices coupled to said SCSI buses and to said first bus means.

19. A digital computer system, including at least a first host processor and a second host processor, a first input/output channel bus coupled to said first host processor, a second input/output channel bus coupled to said second host processor, an interprocessor bus, each of said host processors being coupled together by means of said interprocessor bus, said computer system comprising:

a first device controller, said first device controller having a first port-input/output controller coupled to said first input/output channel bus and a second port-input/output controller coupled to said second input/output channel bus, a first peripheral device-input/output controller coupled to a first SCSI bus and a second peripheral device-input/output controller coupled to a second SCSI bus; and

a second device controller, said second device

controller having a third port-input/output controller coupled to said first input/output channel bus and a fourth port-input/output controller coupled to said second input/output channel bus, a third peripheral device-input/output controller coupled to said first SCSI bus and a fourth peripheral device-input/output controller coupled to said second SCSI bus;

each said port-input/output controller having an ownership latch for granting ownership of each said device controller to multiple host processors and thereby providing at least four data paths between one of said first or second host processors and one of any peripheral devices coupled to said first SCSI bus or said second SCSI bus.

20. The computer system according to claim 19 wherein each said device controller comprises:

a first bus;

a dedicated microprocessor having control over said first bus (the MPU bus);

a second bus, said second bus operative at a different speed than said first bus;

a multiple-channel direct memory access (DMA) controller means comprising a state machine, said DMA controller having control over said second bus (the data buffer bus) and control over and access to said first and second port-input/output controller for arbitrating access to said first bus; and

a first bus switch coupled between said first bus and said second bus for communicating information on said first bus at a first speed to said second bus at a second speed and information on said second bus at said second speed to said first bus at said first speed under control of said DMA controller and upon request of said MPU.

21. For use in a digital computer system, a method for controlling data and command transfer in a device controller comprising:

granting ownership of said device controller to a first host processor by means of a first ownership latch means of a first port input-output controller; and

granting ownership of said device controller to a second host processor independently of said first port input/output controller, by means of a second ownership latch means operative independently of said first ownership latch means thereby to provide multiple data paths between one of said first or second host processors and any peripheral devices coupled to said device controller.

22. The method according to claim 21 further including the steps of:

defining by alternating time allocation a first channel and a second channel on a synchronous bus; and

arbitrating time allocation between said first channel and said second channel, by means of a programmable state machine to regulate data transfer on said synchronous bus, and for allocating time on said synchronous bus to said microprocessor means at a lower priority than for said data transfer.

23. The method according to claim 23 further including the steps of:

imposing a maximum latency between a data request and a data acknowledge issued by said state machine of no more than two bus cycles on any channel and wherein maximum latency between a bus request and a bus grant issued by said state machine is dependent upon load of said first channel and said second channel.

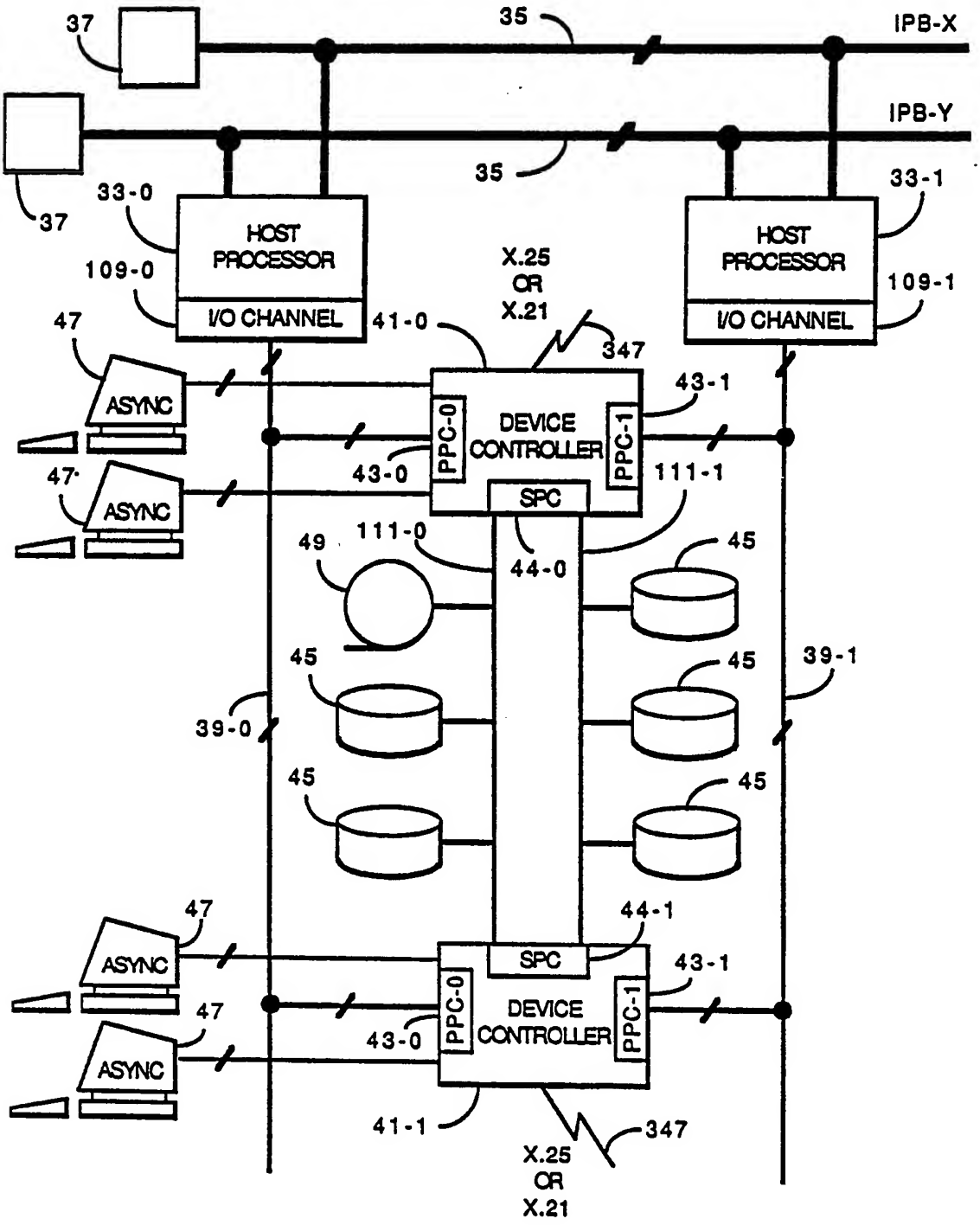
37
35
IPB-X
IPB-Y
35
37
33-0
HOST PROCESSOR
I/O CHANNEL
109-0
X.25 OR X.21
41-0
347
HOST PROCESSOR
I/O CHANNEL
33-1
109-1
47
ASYNC
47
ASYNC
PPC-0
DEVICE CONTROLLER
PPC-1
43-0
43-1
SPC
111-1
111-0
49
44-0
45
45
45
45
45
39-0
39-1
47
ASYNC
47
ASYNC
SPC
44-1
PPC-0
DEVICE CONTROLLER
PPC-1
43-1
43-0
41-1
347
X.25 OR X.21

FIG. 1

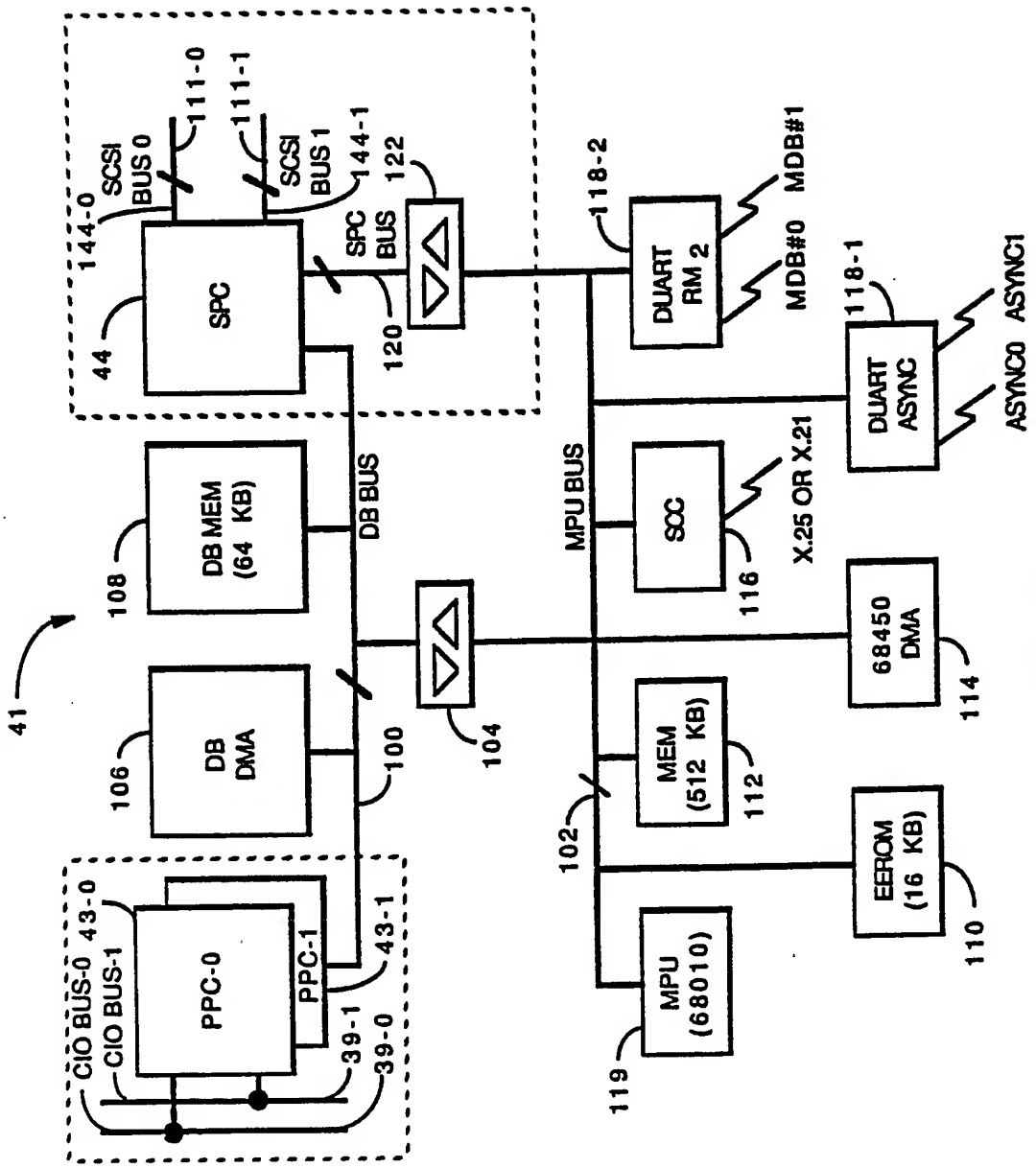
41
SPC
44
144-0
SCSI BUS 0
111-0
111-1
SCSI BUS 1
144-1
SPC BUS
122
120
DB MEM (64 KB)
108
DB DMA
106
DB BUS
100
104
MPU BUS
DUART RM 2
118-2
MDB#0
MDB#1
DUART ASYNC
118-1
ASYNC0
ASYNC1
SCC
116
X.25 OR X.21
68450 DMA
114
MEM (512 KB)
112
102
EEROM (16 KB)
110
MPU (68010)
119
CIO BUS-0
43-0
CIO BUS-1
PPC-0
PPC-1
43-1
39-1
39-0

FIG. 2

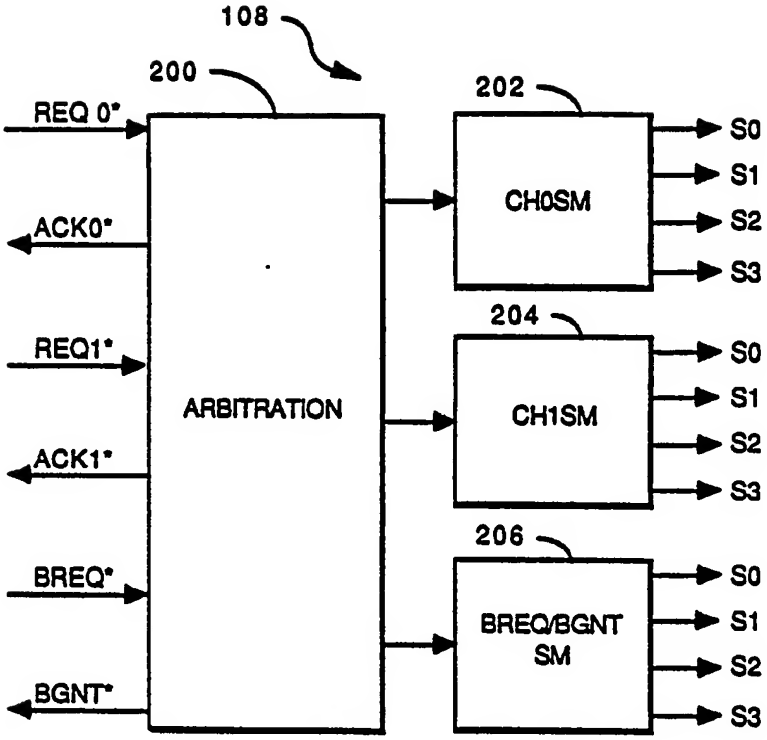
108
200
202
REQ 0*
ACK0*
REQ1*
ACK1*
BREQ*
BGNT*
ARBITRATION
CH0SM
204
CH1SM
206
BREQ/BGNT SM
S0
S1
S2
S3
S0
S1
S2
S3
S0
S1
S2
S3

FIG. 3

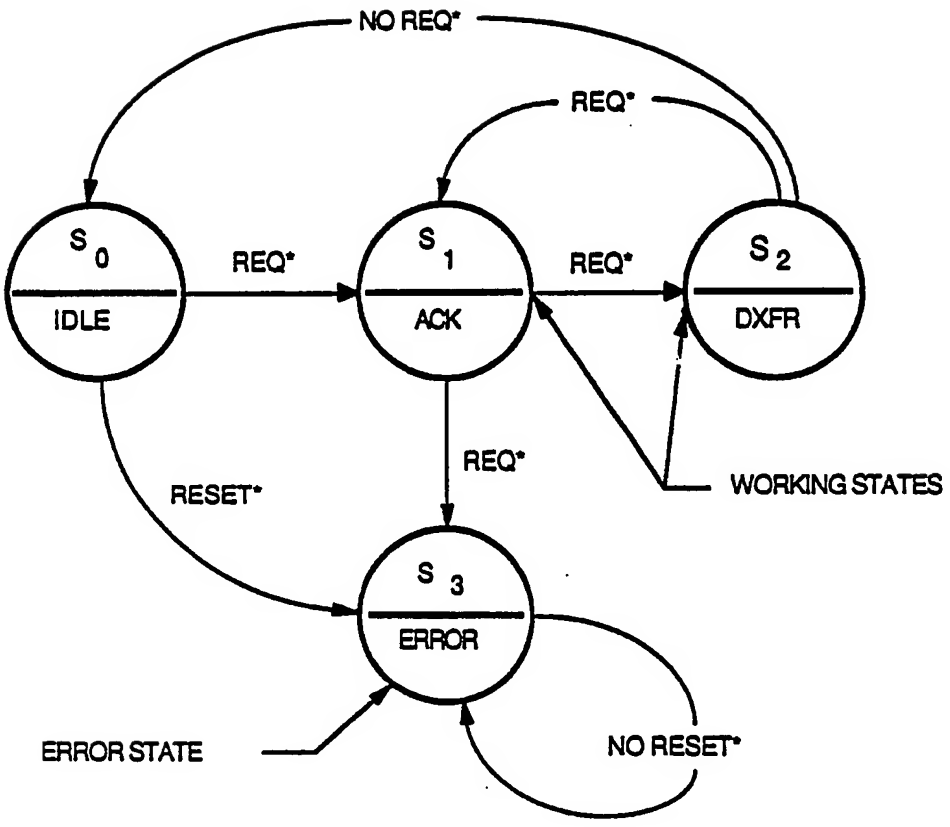
NO REQ*
REQ*
S 0
IDLE
REQ*
S 1
ACK
REQ*
S 2
DXFR
REQ*
WORKING STATES
RESET*
S 3
ERROR
ERROR STATE
NO RESET*

FIG. 4

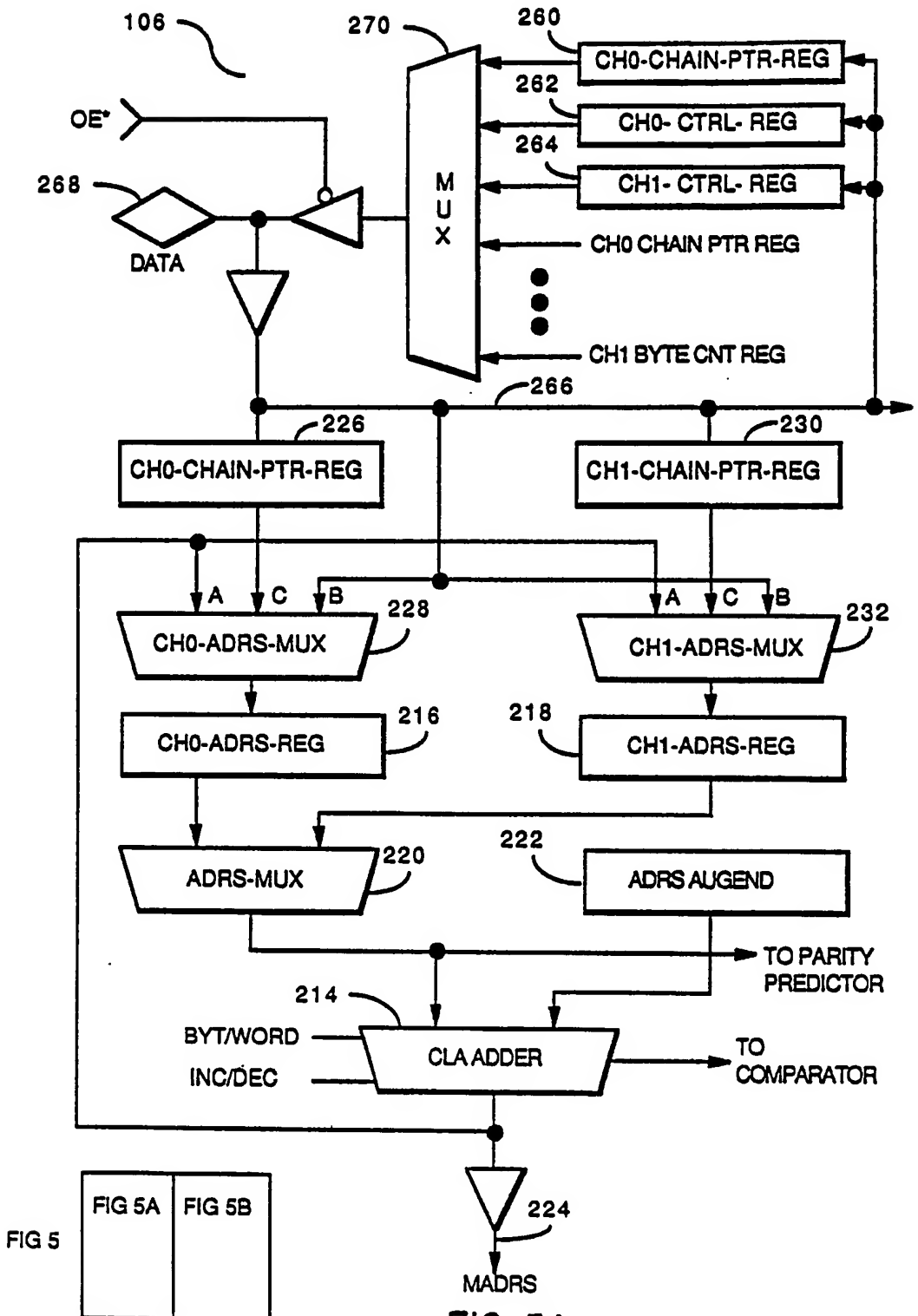

FIG 5

FIG 5A	FIG 5B

FIG. 5A

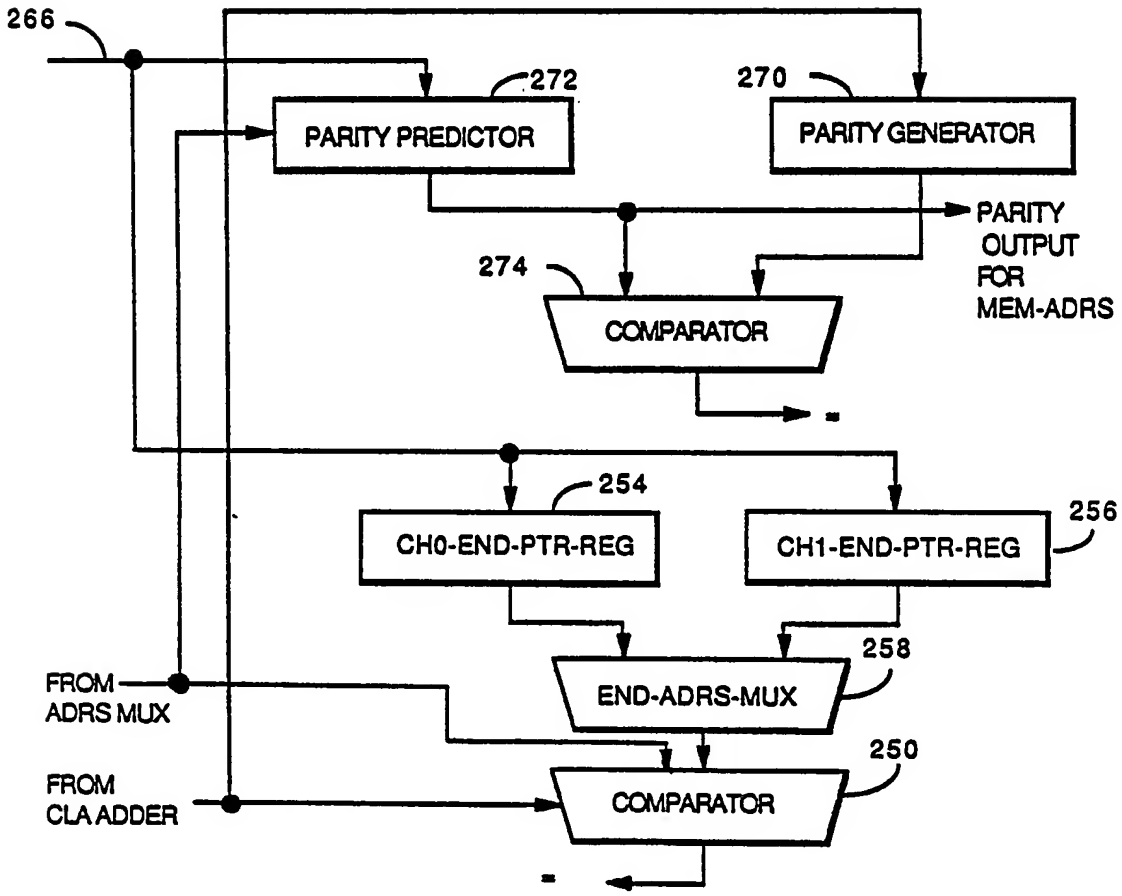
266
272
PARITY PREDICTOR
270
PARITY GENERATOR
PARITY OUTPUT FOR MEM-ADRS
274
COMPARATOR
254
CH0-END-PTR-REG
256
CH1-END-PTR-REG
258
END-ADRS-MUX
FROM ADRS MUX
FROM CLA ADDER
250
COMPARATOR

FIG. 5B

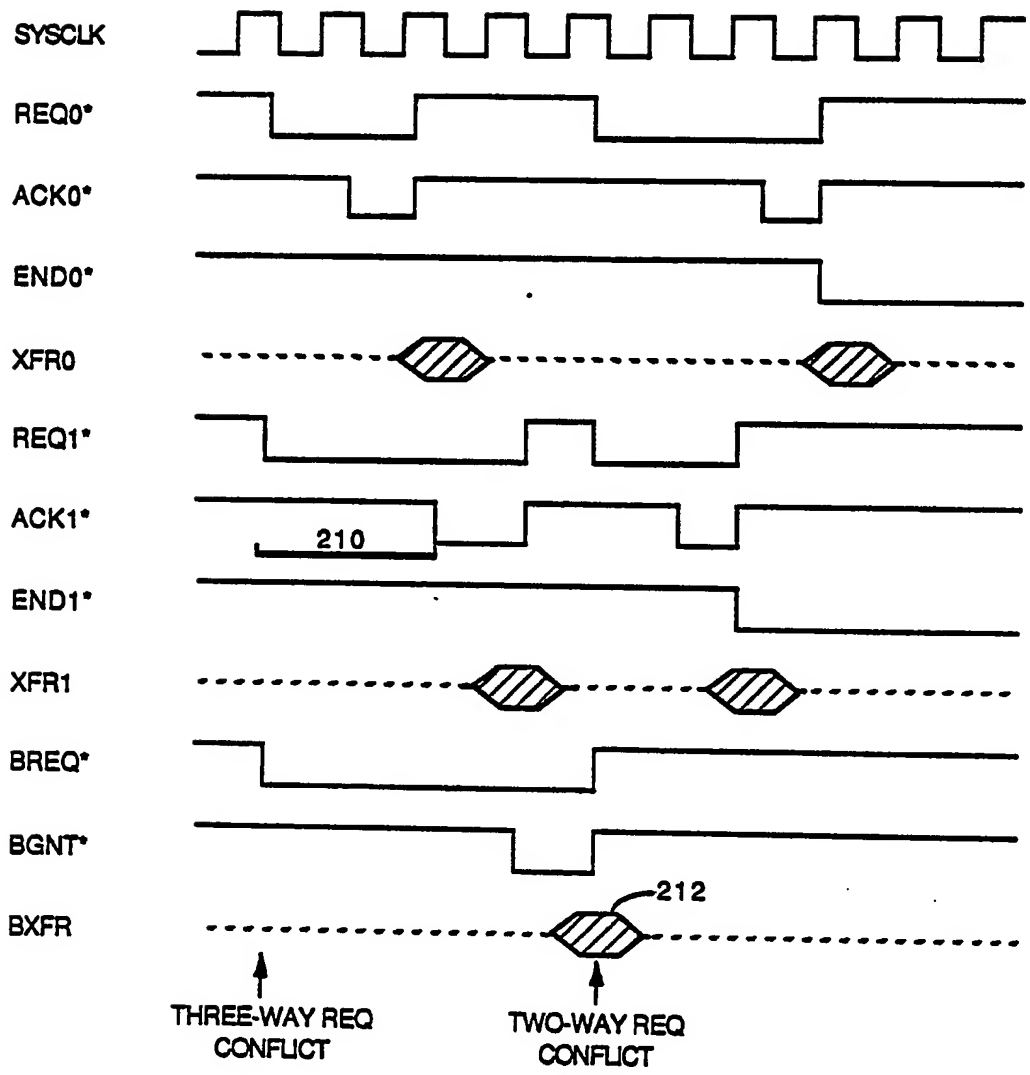
SYSCLK
REQ0*
ACK0*
END0*
XFR0
REQ1*
ACK1*
210
END1*
XFR1
BREQ*
BGNT*
212
BXFR
THREE-WAY REQ CONFLICT
TWO-WAY REQ CONFLICT

FIG. 6

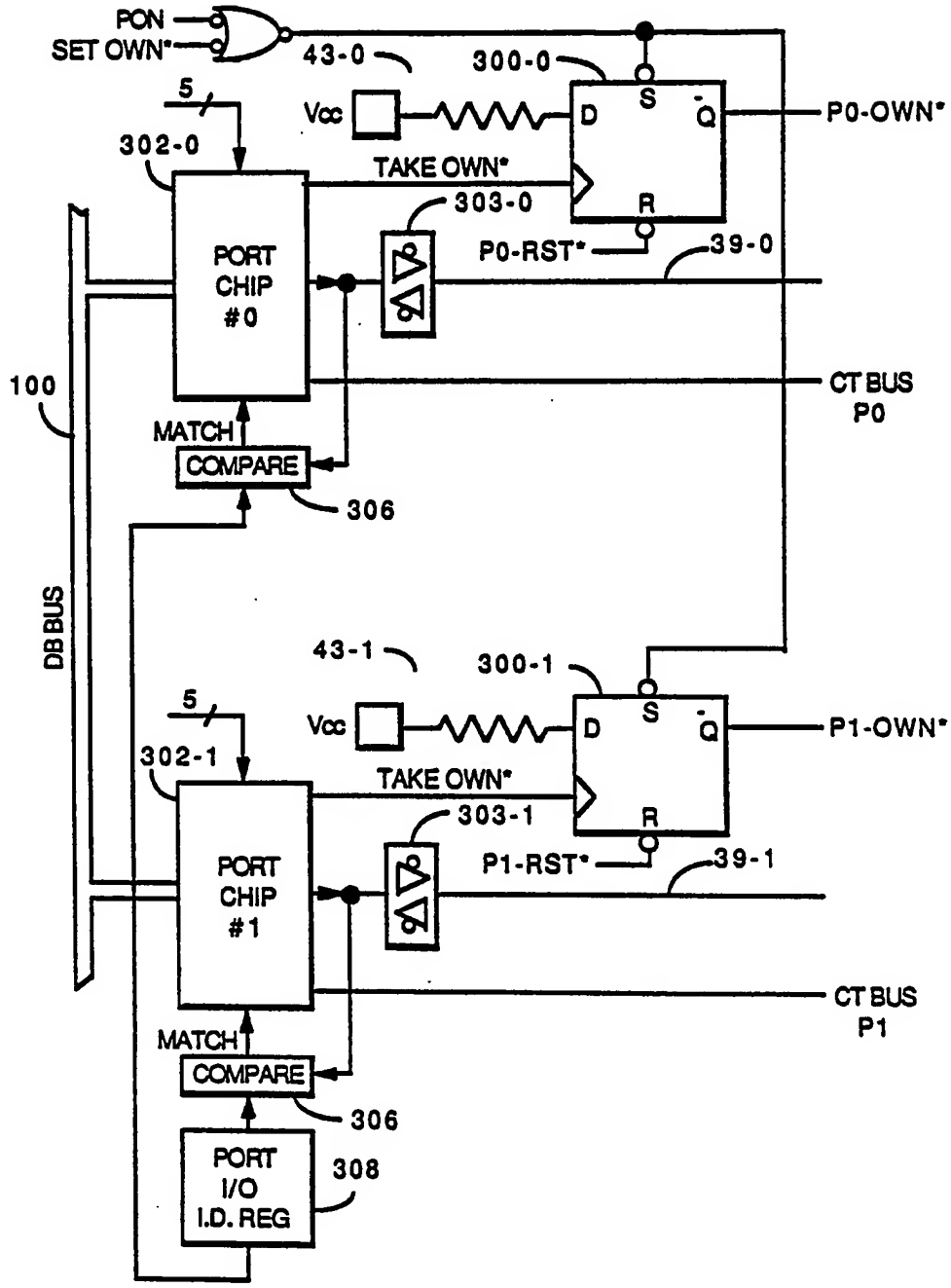
PON
SET OWN*
43-0
300-0
5
Vcc
D
S
Q
R
P0-OWN*
302-0
TAKE OWN*
303-0
PORT CHIP #0
P0-RST*
39-0
100
CT BUS P0
MATCH
COMPARE
306
DB BUS
43-1
300-1
5
Vcc
P1-OWN*
302-1
TAKE OWN*
303-1
PORT CHIP #1
P1-RST*
39-1
CT BUS P1
MATCH
COMPARE
306
PORT I/O I.D. REG
308

FIG. 7

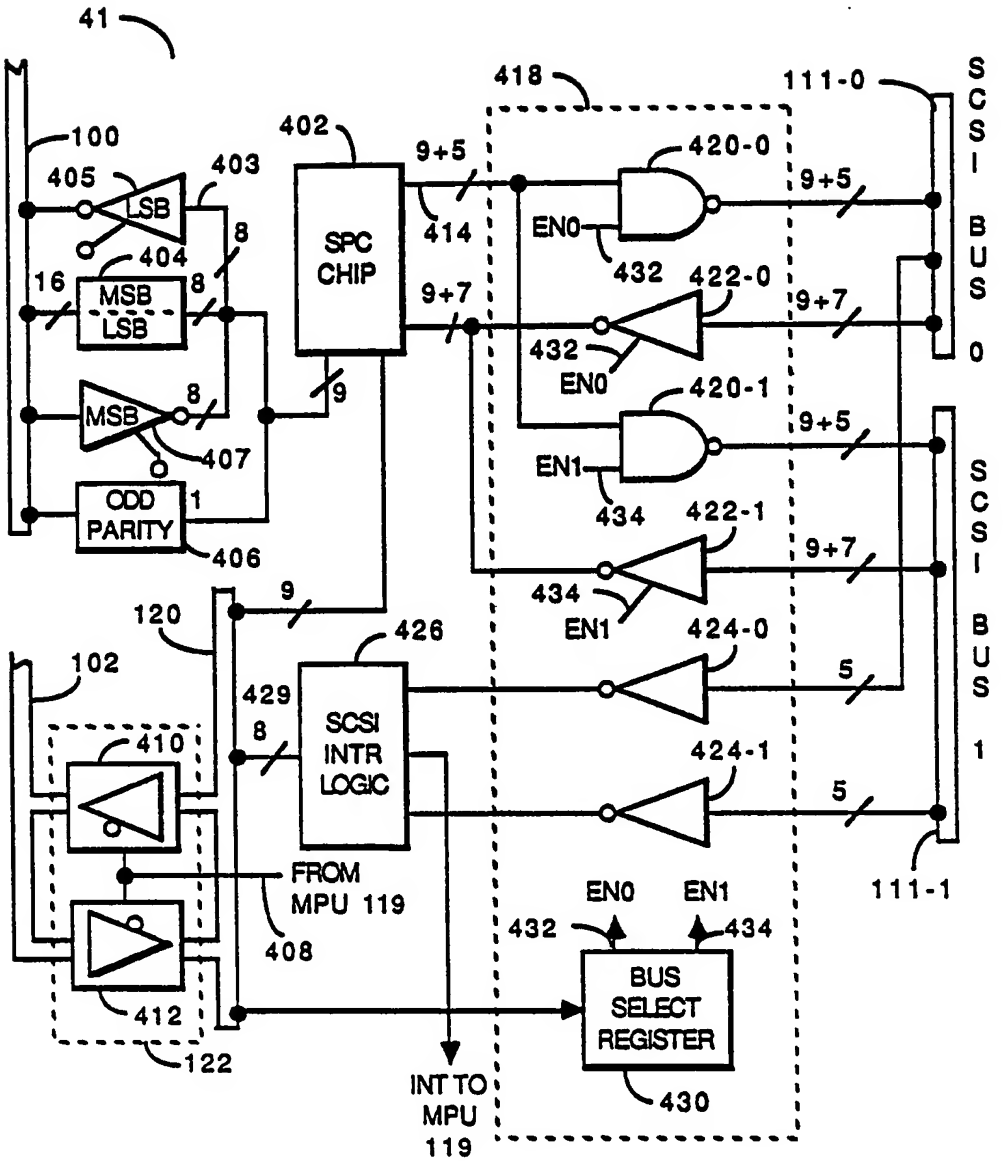
41
100
405
LSB
403
8
404
16
MSB
LSB
8
MSB
8
407
9
ODD
PARITY
1
406
402
SPC
CHIP
9+5
414
9+7
418
420-0
EN0
432
9+5
111-0
422-0
9+7
432
EN0
420-1
EN1
434
9+5
422-1
9+7
434
EN1
424-0
5
424-1
5
SCSI BUS 0
SCSI BUS 1
111-1
120
9
102
426
429
8
SCSI
INTR
LOGIC
410
412
122
FROM
MPU 119
408
INT TO
MPU
119
EN0
EN1
432
434
BUS
SELECT
REGISTER
430

FIG. 8

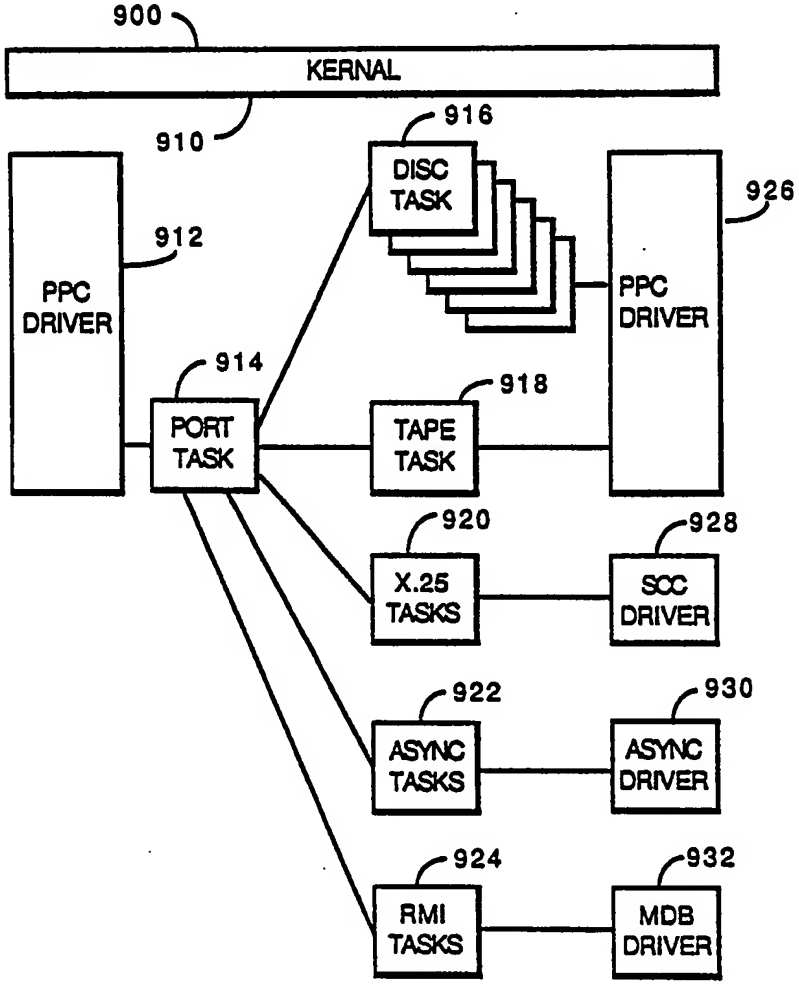
900
KERNAL
910
916
DISC TASK
926
912
PPC DRIVER
PPC DRIVER
914
PORT TASK
918
TAPE TASK
920
928
X.25 TASKS
SCC DRIVER
922
930
ASYNC TASKS
ASYNC DRIVER
924
932
RMI TASKS
MDB DRIVER

FIG. 9

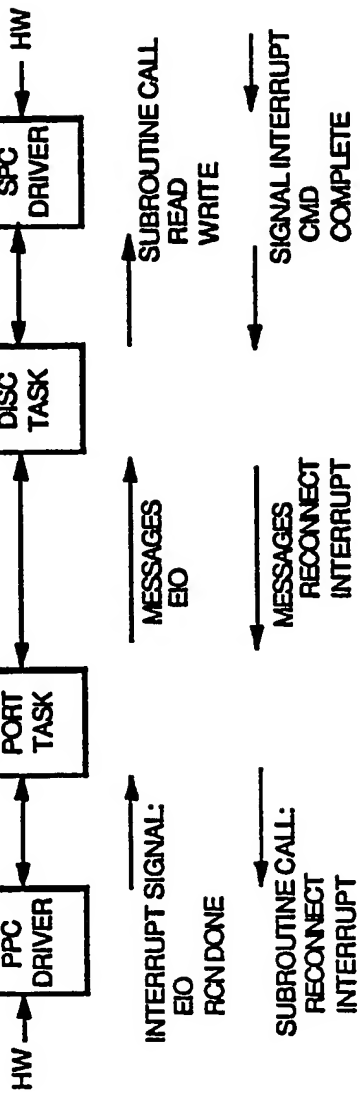
HW
912
PPC DRIVER
914
PORT TASK
916
DISC TASK
926
SPC DRIVER
HW
INTERRUPT SIGNAL:
EIO
RCN DONE
SUBROUTINE CALL:
RECONNECT
INTERRUPT
MESSAGES
EIO
MESSAGES
RECONNECT
INTERRUPT
SUBROUTINE CALL
READ
WRITE
SIGNAL INTERRUPT
CMD
COMPLETE

FIG. 10

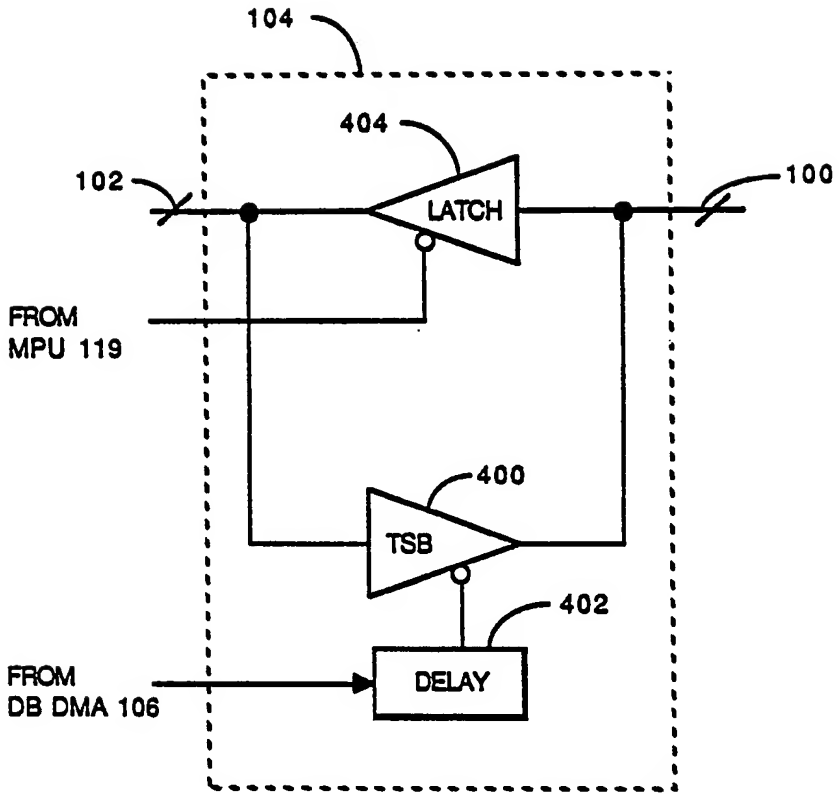
104
404
102
100
LATCH
FROM
MPU 119
400
TSB
402
FROM
DB DMA 106
DELAY

FIG. 11